*CTF 2022 Reverse Writeup

原创

1mmorta1 于 2022-04-17 09:00:00 发布 127 收藏 2
分类专栏： reverse 文章标签： python 系统安全 安全

reverse 专栏收录该内容
12 篇文章 0 订阅
订阅专栏

> AAA : immortal

NaCl

感觉和native client 关系不大 就是mmap出来一个页当成栈来使用, 并且栈顶指针变成了r15而非rsp

关键函数做了反调试, 在加密输入之前会先计算dword_80AFB60 作为key, 具体操作没仔细看反正dump出来就完事了

剩下的就是动调跟着汇编指令慢慢撸逻辑, 加密部分就是一个简单的费斯妥密码后面接一个xtea , 每八个字节都是独立计算的

```python
cmp = [0x66, 0xC2, 0xF5, 0xFD, 0x86, 0x82, 0x32, 0x7A, 0x04, 0x40, 0x94, 0xCE, 0xDC, 0x8A, 0xE0, 0x5D, 0x0A, 0xBD, 0xE4, 0xA6, 0xDC, 0xAD, 0xCA, 0x16, 0x0C, 0x6F, 0xCD, 0x13, 0x36, 0xD9, 0x75, 0x1A]
dword_80AFB60 = [0x4050607, 0x10203, 0x0C0D0E0F, 0x8090A0B, 0x0CD3FE81B, 0x0D7C45477, 0x9F3E9236, 0x107F187, 0x0F993CB81, 0x0BF74166C, 0x0DA198427, 0x1A05ABFF, 0x9307E5E4, 0x0CB8B0E45, 0x306DF7F5, 0x0AD300197, 0x0AA86B056, 0x449263BA, 0x3FA4401B, 0x1E41F917, 0x0C6CB1E7D, 0x18EB0D7A, 0x0D4EC4800, 0x0B486F92B, 0x8737F9F3, 0x765E3D25, 0x0DB3D3537, 0x0EE44552B, 0x11D0C94C, 0x9B605BCB, 0x903B98B3, 0x24C2EEA3, 0x896E10A2, 0x2247F0C0, 0x0B84E5CAA, 0x8D2C04F0, 0x3BC7842C, 0x1A50D606, 0x49A1917C, 0x7E1CB50C, 0x0FC27B826, 0x5FDDDFBC, 0x0DE0FC404, 0x0B2B30907]
xtea_key = [0x3020100,0x7060504,0x0b0a0908,0x0f0e0d0c]
rol = lambda val, r_bits, max_bits: \
    (val << r_bits%max_bits) & (2**max_bits-1) | \
    ((val & (2**max_bits-1)) >> (max_bits-(r_bits%max_bits)))
ror = lambda val, r_bits, max_bits: \
    ((val & (2**max_bits-1)) >> r_bits%max_bits) | \
    (val << (max_bits-(r_bits%max_bits)) & (2**max_bits-1))
def xtea_encrypt(rounds, v, k):
    v0 = v[0]
    v1 = v[1]
    x = 0
    delta = 0x10325476
    for i in range(rounds):
        v0 += (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (x + k[x & 3])
        v0 = v0 & 0xFFFFFFFF
        x += delta
        x = x & 0xFFFFFFFF
        v1 += (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (x + k[(x >> 11) & 3])
        v1 = v1 & 0xFFFFFFFF
    v[0] = v0
    v[1] = v1
    return v
def xtea_decrypt(rounds, v, k):
```

```
def xtea_decrypt(rounds, v, k):
    v0 = v[0]
    v1 = v[1]
    delta = 0x10325476
    x = delta * rounds
    for i in range(rounds):
        v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (x + k[(x >> 11) & 3])
        v1 = v1 & 0xFFFFFFFF
        x -= delta
        x = x & 0xFFFFFFFF
        v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (x + k[x & 3])
        v0 = v0 & 0xFFFFFFFF
    v[0] = v0
    v[1] = v1
    return v

def crypt(plain):
    res = b''
    for ii in range(4):
        left = int.from_bytes(plain[ii*8:ii*8+4],'big')
        right= int.from_bytes(plain[ii*8+4:ii*8+8],'big')
        for i in range(0x2c):
            ebx = rol(left,1,32) & rol(left,8,32)
            ebx ^= rol(left,2,32)
            ebx ^= right
            ebx ^= dword_80AFB60[i]
            right = left
            left = ebx
        print(hex(left),hex(right))
        left, right = xtea_encrypt(2**(ii+1),[right,left],xtea_key)
        res += left.to_bytes(4,'little')+right.to_bytes(4,'little')
    return res

def decrypt(cipher):
    res = b''
    for ii in range(4):
        left = int.from_bytes(cipher[ii*8:ii*8+4],'little')
        right= int.from_bytes(cipher[ii*8+4:ii*8+8],'little')
        right, left = xtea_decrypt(2**(ii+1),[left,right],xtea_key)
        for i in range(0x2b,-1,-1):
            ebx = rol(right,1,32) & rol(right,8,32)
            ebx ^= rol(right,2,32)
            ebx ^= dword_80AFB60[i]
            last_right = ebx ^ left
            left = right
            right = last_right
        res += left.to_bytes(4,'big')+right.to_bytes(4,'big')
    return res

print(decrypt(cmp))
```

Jump

首先因为是静态链接的文件,所以先自己编写了一个c语言文件用bindiff恢复一些符号, 发现其中大量调用了setjmp,联想题目名字知道这是关键函数.

所以搜索一下,找到Linux manual : https://linux.die.net/man/3/sigsetjmp https://linux.die.net/man/3/longjmp val = setjmp(env)存下context, 并且val值为0. 然后每次longjmp(env, val) 就跳转到上一次执行setjmp(env)的位置并且此时setjmp(env)处的val实际上为longjmp传入的val(如果val设置为0会自动被改为1). 程序实际上利用这个跳转实现了循环, val作为循环变量i.

再自己实现一个longjmp函数编译出文件以后bindiff, 恢复出源文件里的longjmp函数.

程序本身的逻辑很简单, 在通过输入生成了0x22个字符串以后, 通过sub_401F62对字符串的首字节进行递增的排序,然后由于比较的是字符串的最后一个数据,因此可以恢复出原来的flag.

```
res = [ 3, 106, 109,  71, 110,  95,  61, 117,  97,  83, 90,  76, 118,  78,  52, 119,  70, 120,  69,  54, 82,  43
, 112,   2,  68,  50, 113,  86,  49,  67, 66,  84,  99, 107]
q = res.copy()
t = sorted(res)
cur = q.index(2)
ans = []
while q[cur]!=3:
    nxt = t[cur]
    ans.append(nxt)
    cur = q.index(nxt)

print(bytes(ans))
```